First Name	**Last Name**	**First letter of last name**
UW Userid		

ECE 327 Midterm

2014t1 (Winter)

Instructions and General Information

- 100 marks total
- Time limit: 1 hour and 20 minutes (80 minutes)
- No books, no notes, no computers.   Calculators are allowed
- If you need extra paper, request some from a proctor.
- Write neatly.
- To earn part marks, you must show the formulas you use and all of your work.
- **The proctors and instructors will not answer questions, except in cases where an error on the exam is suspected. If you are confused about a question, write down your assumptions or interpretation.**
- **Justifications of answers will be marked according to correctness, clarity, and concision.**

			Total Marks	**Approx. Time**	**Page**
Q0	!!Almost Free!!		2	2	2
Q1	Simulation		20	15	3
Q2	The Gold, the Silver, and the Bronze		17	10	6
Q3	State Machine Behaviour		18	10	8
Q4	Allocation and Control		15	15	9
Q5	Bubbles and Machines		10	7	12
Q6	Design with Memory		20	15	13
Totals			100	74	

Q0 (2 Marks) !!Almost Free!!

(*estimated time: 2 minutes*)

Q0a (1 Mark) Best part

What is the best part of the course?

Q0b (1 Mark) Most improve

What one thing could be done to most improve the course for the remainder of the term?

Q1 (20 Marks) Simulation

(*estimated time: 15 minutes*)

For the code fragment on the next page, *briefly* describe what happens in each *simulation cycle*, beginning at 3 ns.

NOTES:

1. End your simulation after seven simulation cycles or just before 10 ns, whichever occurs first.
2. If you need less than seven simulation cycles, then write "N/A" in each simulation cycle that you do not need.
3. If, after seven simulation cycles, the simulation has not reached 10 ns, then write "$\surd$" in the box for *ran out of simulation cycles*.
4. All signals are `std_logic`.
5. Just before 3 ns, the signals have the values shown in the waveform for scratch work on the next page.

Waveform for scratch work

3ns
a 0
b 0
c 1
d 0
e 1

Answer:

1. ______

2. ______

3. ______

4. ______

5. ______

6. ______

7. ______

```
p_a : process begin
  a <= '0';
  wait for 3 ns;
  a <= '1';
  wait for 7 ns;
  a <= '0';
  wait;
end process;

p_b : process (a, e) begin
  b <= a and e;
end process;

p_c : process (a) begin
  c <= not a;
end process;

p_d : process (c) begin
  d <= not c;
end process;

p_e : process (b, d) begin
  e <= not( b ) or d;
end process;
```

☐ **Ran out of simulation cycles**

The next page is also for scratch work or your answer

Extra copy for scratch work

```
p_a : process begin
  a <= ’0’;
  wait for 3 ns;
  a <= ’1’;
  wait for 7 ns;
  a <= ’0’;
  wait;
end process;

p_b : process (a, e) begin
  b <= a and e;
end process;

p_c : process (a) begin
  c <= not a;
end process;

p_d : process (c) begin
  d <= not c;
end process;

p_e : process (b, d) begin
  e <= not( b ) or d;
end process;
```

Q2 (17 Marks) The Gold, the Silver, and the Bronze

(*estimated time: 10 minutes*)

For each of the code fragments Q2a–Q2b:

1. Answer whether the code is *legal*
2. If the code is *illegal*: explain why, and proceed to the next code fragment.
3. Answer whether the code is *synthesizable*.
4. If the code is *unsynthesizable*: explain why, and proceed to the next code fragment.
5. Answer whether the code adheres to good coding practices, according to the guidelines for ECE 327.
6. If the code does *not follow good coding practices*: explain why.

NOTES:

1. If the code is synthesizable: draw the circuit that would most likely result from synthesizing the code.
2. If the VHDL code includes an implicit state machine: draw the gates, wires, and flops for the datapath. All of the arithmetic and logical operators in the VHDL code (*e.g.*, "+", "−", "<", and "`xor`") are considered part of the datapath.
3. You may draw the control portion of the circuit as a cloud or black-box that drives the appropriate signals in the datapath.

Q2a

```
signal m : std_logic( 0 to 3 );
signal p : std_logic( 1 to 3 );
...
process (m, p)
begin
  p(0) <= '0';
  for i in 1 to 3 loop
    p(i) <= p(i-1) xor m(i);
  end loop;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable or bad practice:

Drawing of hardware, if synthesizable:

Q2b

```
signal m : unsigned( 7 downto 0 );
...
process begin
  wait until rising_edge(clk);
  m <= to_unsigned( 0, 8 );
  wait until rising_edge(clk);
  while m < 3 loop
    m <= m + 1;
    wait until rising_edge(clk);
  end loop;
end process;
```

	Yes	No
Legal	☐	☐
Synthesizable	☐	☐
Good Practice	☐	☐

Explanation if illegal, unsynthesizable or bad practice:

Drawing of hardware, if synthesizable:

Q3 (18 Marks) State Machine Behaviour

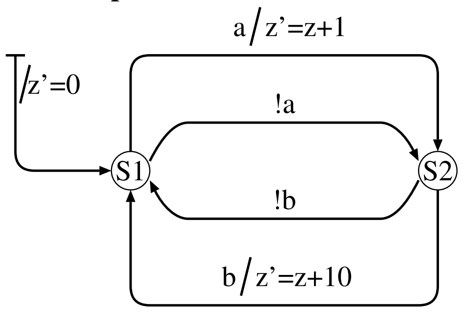
(*estimated time: 10 minutes*)

In this question, you will compare the behaviour of z in the state machines Q3a–Q3b against z in the specification machine.

NOTES:

1. A state machine is *correct* if its z has the *same sequence* of values as z in the specification machine.
2. Ignore any differences in the first few clock cycles.
3. The value on z does *not* need to appear at the same time as in the specification machine. Correctness is determined only by the sequence of values; a *time-shift* between the state machine and the specification machine is allowed. (Illustrations of "time-shift" are given below.)
4. If the state machine is *correct*, answer what the *time-shift* is from the specification machine.
5. If the state machine is *incorrect*, explain either how the machines behaviour differs from the specification machine or how the state machine could be modified to fix the incorrect behaviour.

Specification machine



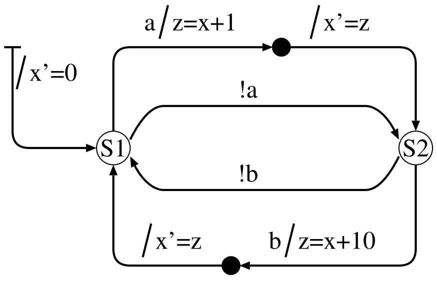
Time-shift examples

(waveform values are just examples, they do not match the actual values of z in the specification machine)

specification z	1	3	6	7	8	9
time-shift = 1	• • • •	1	3	6	7	8

specification z	1	3	6	7	8	9
time-shift = -2	6	7	8	9	• • • • • • • •	

Q3a

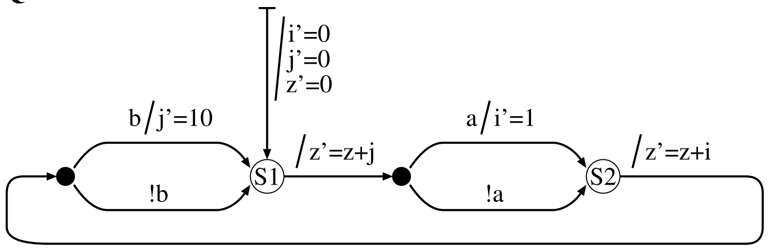


	Yes	No
Correct behaviour	☐	☐

If correct: time-shift= ☐

If incorrect, explanation or modification:

Q3b



	Yes	No
Correct behaviour	☐	☐

If correct: time-shift= ☐

If incorrect, explanation or modification:

Q4 (15 Marks) Allocation and Control

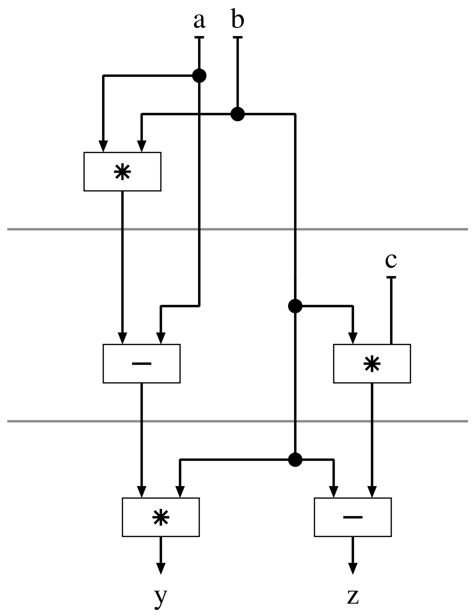
(*estimated time: 15 minutes*)

For the dataflow diagram below: perform input/output allocation, datapath allocation, and register allocation; then draw the control table.

NOTES:

1. You may use 2:1 multiplexers, and may combine 2:1 multiplexers to create larger multiplexers.
2. The optimization goals, in order of highest priority to lowest, are to minimize the number of:
   (a) multipliers and subtracters
   (b) input and output ports
   (c) registers
   (d) 2:1 multiplexers on *registers* (including the 2:1 multiplexers used to build larger multiplexers)
   (e) 2:1 multiplexers on *multipliers and subtracters* (including the 2:1 multiplexers used to build larger multiplexers)
   (f) chip-enables
3. You may *not* perform scheduling optimizations on the dataflow diagram.
4. The *only* algebraic optimization that you may perform is *commutativity*.

Q4a (7 Marks) Allocation



Number of multiplexers on:

Registers	
Multipliers	
Subtracters	

Q4b (8 Marks) Control Table and Multiplexer Count

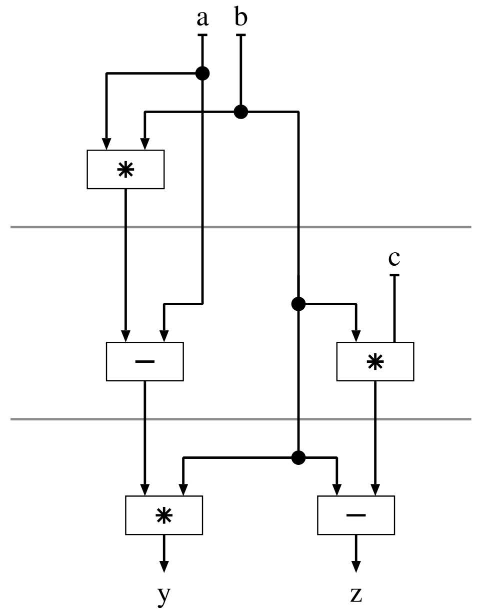
Draw the control table for the dataflow diagram and calculate the number of multiplexers.

NOTES:

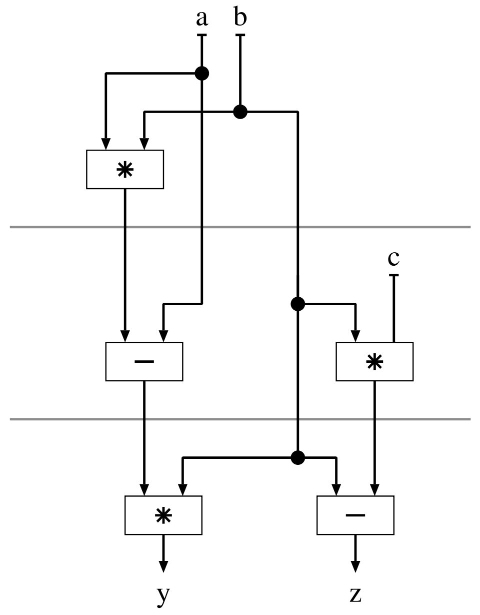
1. Label each row and column of the table clearly.
2. The system shall support an ASAP parcel schedule between parcels.
3. **Clearly show how you calculated the number of multiplexers.**

There are extra copies of the dataflow diagram and space for control tables on the next page.

Extra copy for scratch work



Extra copy for scratch work



Q5 (10 Marks) Bubbles and Machines

(*estimated time: 7 minutes*)

Draw the state-machine diagram for the pseudocode specification below, where the "`#---`" comments specify the clock-cycle boundaries.

NOTES:

1. The system shall support an *indeterminate number of bubbles* between parcels.

```
#------------------
r1 = i1 + i2;
#------------------
if r1 > 23 then {
  m = r1 - i2;
} else {
  m = r1 + i2;
}
r1 <= m + 5;
#------------------
o1 <= r1;
```

Q6 (20 Marks) Design with Memory

(*estimated time: 15 minutes*)

Draw a dataflow diagram that implements the pseudocode specification:

```
M[a-1] = b;
M[c]   = d;
z      = M[a] + M[c];
```

NOTES:

1. It is guaranteed that $a < c$.
2. Inputs shall be *registered*
3. Outputs may be *either combinational or registered*
4. The system shall support an *indeterminate number of bubbles*
5. Memory has registered inputs and combinational outputs (same as in class)
6. The memory shall be *dual-ported*. If you do not know how to answer the question with dual-ported memory, you may earn *part marks* by writing a $\surd$ in the box at the end of this note and answering the question for *single-ported memory*.

   I am answering for *single-ported memory*, not dual-ported memory. ☐
7. Optimization goals in order of decreasing importance:
   (a) minimize *latency*
   (b) minimize *clock period*
   (c) minimize *area*
      i. input ports
      ii. adders and subtracters
      iii. registers (*excluding* memory)
      iv. output ports
8. Input values may be read in any clock cycle, but each input value shall be read exactly once.
9. Optimizations to the pseudocode are allowed, as long as the final values of z and M are correct.

The next page is also for scratch work or your answer

Additional copy of specification pseudocode:

```
M[a-1] = b;
M[c]   = d;
z      = M[a] + M[c];
```